



LẬP TRÌNH ANDROID

Module 3

☞ Click vào phụ lục để chuyển tới bài cần đọc

Phụ lục

Bài 1 Content Provider	2
Bài 2 SQLite	12
Bài 3 Truy vấn và sắp xếp dữ liệu	18
Bài 4 Menu	24
Bài 5 Action Bar	35
Bài 6 Animation	47
Bài 7 Cursor Loader	60
Bài 8 AsyncTask – Thread & Handler.....	70
Bài 9 Broadcast Receiver & Service	76



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

Lập trình Android

Bài 1. Content Provider

Ngành Mạng & Thiết bị di động

cuu duong than cong . com
2014



5014



cuu duong than cong . com

Nội dung



1. Các khái niệm cơ bản

- Content Provider
- Content URI
- ContentResolver

2. Xây dựng ContentProvider cho ứng dụng



1.1 Content Provider



❑ Content Provider:

- Quản lý việc truy nhập dữ liệu chung trên hệ thống từ các ứng dụng khác nhau, gồm một hay nhiều bảng dữ liệu.
- Thể hiện dữ liệu ở dạng bảng quan hệ:
 - Cột: thể hiện trường dữ liệu (hoặc thuộc tính dữ liệu)
 - Dòng: một thể hiện dữ liệu

word	app id	frequency	locale	_ID
mapreduce	user1	100	en_US	1
precompiler	user14	200	fr_FR	2
applet	user2	225	fr_CA	3
const	user1	255	pt_BR	4
int	user5	100	en_UK	5



1.1 Content Provider



❑ Content Provider:

- ContentProvider được cung cấp sẵn trong hệ thống:
 - Hệ thống báo thức (Alarm)
 - Nội dung trình duyệt (Browser)
 - Danh bạ (Contacts)
 - Lịch (Calendar)
 - Tập tin tài liệu (Document)
 - Tập tin đa truyền thông (Media Store)
 - Tùy chỉnh hệ thống (Setting)
 - Điện thoại (Telephony)
 - Từ điển người dùng (UserDictionary)
 - Hộp thư thoại (VoiceMailContract)



1.2 Content URI



❑ Content URI:

- Đối tượng cho phép định nghĩa cách thức truy xuất dữ liệu trên Content Provider, bao gồm:
 - Authority: Tên đầy đủ của Provider.
 - Path: tên bảng tồn tại trong Provider
- Ví dụ về Content URI truy xuất dữ liệu danh bạ:
 - `ContactsContract.Data.CONTENT_URI`
 - `content://com.android.contact/data`
 - Trong đó:
 - `content: scheme`
 - `com.android.contact: authority`
 - `data: tên bảng`





1.3 ContentResolver

❑ ContentResolver:

- Đối tượng cho phép truy xuất dữ liệu và thao tác trên Content Provider, bao gồm:
 - Query
 - Insert
 - Update
 - Delete
- Truy xuất đối tượng ContentResolver thông qua phương thức:
 - `getContentResolver();`



1.3 ContentResolver

❑ ContentResolver:

- Phương thức Query:
 - Cú pháp:
`query(Uri, projection, selection, selectionArgs, sortOrder)`
 - Uri: đường dẫn truy xuất dữ liệu. - URI
 - Projection: mảng tên các trường dữ liệu (cột) cần truy xuất. – String[]
 - Selection: câu điều kiện truy vấn. - String
 - SelectionArgs: mảng biến tham số cho câu điều kiện truy vấn. – String[]
 - SortOrder: tiêu chí sắp xếp dữ liệu truy vấn (tên cột). - String
 - Kết quả trả về:
 - Cursor: chứa bảng dữ liệu kết quả truy vấn.



1.3 ContentResolver



❑ ContentResolver:

- Phương thức Query:

- Ví dụ truy xuất tất cả dữ liệu danh bạ trên thiết bị:

```
// Định nghĩa đường dẫn truy xuất
```

```
Uri uri = ContactsContract.Data.Content_URI;
```

```
// Định nghĩa tham số truy vấn
```

```
String projection = null; // Lấy ra tất cả các cột
```

```
String selection = null; // Không có điều kiện truy vấn
```

```
String selectioArgs = null; // Không có tham số truy vấn
```

```
String sortOrder = null; // Sắp xếp mặc định
```

```
// Đổi tượng truy vấn
```

```
ContentResolver cr = this.getContentResolver();
```

```
Cursor cursor = cr.query(Uri, projection, selection, selectionArgs, sortOrder);
```



1.3 ContentResolver



❑ ContentResolver:

- Phương thức Query:

- Ví dụ truy xuất dữ liệu danh bạ trên thiết bị có tên bắt đầu bằng "M" và sắp xếp theo ID.

```
// Định nghĩa đường dẫn truy xuất
```

```
Uri uri = ContactsContract.Data.Content_URI;
```

```
// Định nghĩa tham số truy vấn
```

```
String projection = null; // Lấy ra tất cả các cột
```

```
String selection = Data.DISPLAY_NAME + " like ?"; // điều kiện truy vấn
```

```
String selectioArgs = { "M%"}; // Tham số truy vấn
```

```
String sortOrder = Data._ID; // Sắp xếp theo ID
```

```
// Đổi tượng truy vấn
```

```
ContentResolver cr = this.getContentResolver();
```

```
Cursor cursor = cr.query(Uri, projection, selection, selectionArgs, sortOrder);
```



1.3 ContentResolver



❑ ContentResolver:

- Phương thức Insert, Update: được thực hiện thông qua tham số dữ liệu **ContentValues**:
 - **ContentValues**: cho phép định nghĩa dữ liệu khi thao tác, thông qua phương thức **put(Key, Value)**.
 - Trong đó:
 - Key: tên trường dữ liệu (cột). - String
 - Value: dữ liệu tương ứng với trường dữ liệu – String, Float, Long...



1.3 ContentResolver



❑ ContentResolver:

- Phương thức Insert:
 - Cú pháp:

insert(Uri, ContentValues)

 - Uri: đường dẫn truy xuất dữ liệu. - URI
 - ContentValues: dữ liệu cần thêm vào. - ContentValues
 - Kết quả trả về:
 - Uri: đường dẫn đến dữ liệu vừa thêm vào



1.3 ContentResolver



❑ ContentResolver:

- Phương thức Update:

- Cú pháp:

update(Uri, ContentValues, where, selectionArgs)

- Uri: đường dẫn truy xuất dữ liệu. - URI
 - ContentValues: dữ liệu cần cập nhật. - ContentValues
 - Where: điều kiện dữ liệu được cập nhật. - String
 - SelectionArgs: mảng tham số cho câu điều kiện. – String[]

- Kết quả trả về:

- int: số lượng dòng dữ liệu đã được cập nhật



1.3 ContentResolver



❑ ContentResolver:

- Phương thức Delete:

- Cú pháp:

delete(Uri, ContentValues, where, selectionArgs)

- Uri: đường dẫn truy xuất dữ liệu. - URI
 - Where: điều kiện dữ liệu được xóa. - String
 - SelectionArgs: mảng tham số cho câu điều kiện. – String[]

- Kết quả trả về:

- int: số lượng dòng dữ liệu đã được xóa.



Nội dung



1. Các khái niệm cơ bản
2. Xây dựng ContentProvider cho ứng dụng
 - Vấn đề xây dựng ContentProvider
 - Authority & UriMatcher



2.1 Vấn đề xây dựng ContentProvider



- ❑ Việc sử dụng ContentProvider trong ứng dụng khi cần xây dựng những tính năng sau:
 - Cung cấp các dữ liệu hoặc tập tin đặc trưng, phức tạp cho những ứng dụng khác.
 - Cho phép người dùng sao chép các dữ liệu vào các ứng dụng khác.
 - Sử dụng và tùy chỉnh tính năng tìm kiếm dữ liệu trong ứng dụng.



2.1 Vấn đề xây dựng ContentProvider



❑ Thực hiện xây dựng ContentProvider:

- Xây dựng dữ liệu
 - Dữ liệu tập tin (not recommended)
 - Dữ liệu có cấu trúc (SQLite)
- Thiết kế các biến truy xuất dữ liệu (Contract class)
 - Authority – Content URI (required)
 - Biến đại diện cho các trường dữ liệu (tên cột)
- Tạo lớp kế thừa và thực hiện các phương thức truy vấn dữ liệu
 - ContentProvider (required)
 - onCreate
 - query - delete - insert – update
- Khai báo lớp sử dụng trong Manifest thông qua thẻ <provider>
 - Khai báo Permission



2.2 Authority & UriMatcher



❑ Authority:

- Định nghĩa đường dẫn truy xuất vào ContentProvider.
 - Định nghĩa đê xuất: <package_name>.<app_name>.provider
 - Ví dụ:
 - com.htsi.t3h.onlineshop.provider
- Định nghĩa đường dẫn truy xuất vào từng bảng riêng biệt trong ContentProvider.
 - Định nghĩa đê xuất: <authority>/<table_name>
 - Ví dụ:
 - com.htsi.t3h.onlineshop.provider/accessory
 - com.htsi.t3h.onlineshop.provider/clothes





2.2 Authority & UriMatcher

□ UriMatcher:

- Đối tượng cho phép kiểm tra các Content URI truy xuất vào Content Provider.
 - Sử dụng dấu đại diện:
 - * : Cho phép truy xuất vào tất cả các dạng dữ liệu.
 - # : Cho phép truy xuất vào một dòng trong một bảng.
 - Ví dụ định nghĩa Content URI truy xuất dữ liệu:
 - Truy xuất tất cả các bảng trong ContentProvider
 - content://com.htsi.t3h.onlineshop.provider/*
 - Truy xuất tất cả các dữ liệu trong bảng accessory
 - content://com.htsi.t3h.onlineshop.provider/accessory/*
 - Truy xuất dòng dữ liệu có id = # trong bảng accessory
 - content://com.htsi.t3h.onlineshop.provider/accessory/#



Thảo luận





Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

Lập trình Android

Bài 2. *SQLite*

Ngành Mạng & Thiết bị di động

cuu duong than cong . com

2014



2014



cuu duong than cong . com

Nội dung



1. Các khái niệm cơ bản
2. Xây dựng CSDL với SQLite



1. Các khái niệm cơ bản



❑ SQLite:

- Thể hiện dữ liệu ở dạng bảng quan hệ:
 - Cột: thể hiện trường dữ liệu (hoặc thuộc tính dữ liệu)
 - Dòng: một thể hiện dữ liệu

word	app id	frequency	locale	_ID
mapreduce	user1	100	en_US	1
precompiler	user14	200	fr_FR	2
applet	user2	225	fr_CA	3
const	user1	255	pt_BR	4
int	user5	100	en_UK	5





1. Các khái niệm cơ bản

- ❑ SQLite là cơ sở dữ liệu mở được nhúng vào Android, hỗ trợ các đặc điểm về quan hệ chuẩn của cơ sở dữ liệu như cú pháp, transaction, các câu lệnh.
- ❑ Sử dụng SQLite không yêu cầu về thiết lập bất cứ cơ sở dữ liệu hoặc đòi hỏi quyền admin.
- ❑ Hỗ trợ các kiểu dữ liệu : TEXT, INTEGER, REAL, BLOB.



1. Các khái niệm cơ bản

- ❑ Đường dẫn của cơ sở dữ liệu:
 - DATA/data/APP_NAME/databases/FILENAME
- ❑ Cơ sở dữ liệu SQLite cung cấp riêng cho ứng dụng tạo nó.
Nếu cần cung cấp dữ liệu cho các ứng dụng khác, nên sử dụng Content Provider.





2. Xây dựng CSDL với SQLite

❑ Các bước khởi tạo cơ sở dữ liệu với SQLite:

- Tạo lớp kế thừa từ lớp SQLiteOpenHelper.
- Override phương thức onCreate() để tạo cơ sở dữ liệu.
- Override phương thức onUpgrade() để nâng cấp cơ sở dữ liệu.
- Lớp SQLiteOpenHelper cung cấp 2 phương thức `getReadableDatabase()` và `getWritableDatabase()` để sử dụng.
- Tạo đối tượng SQLiteDatabase để truy xuất dữ liệu.



2. Xây dựng CSDL với SQLite

❑ Đối tượng SQLiteOpenHelper:

- Thực hiện các phương thức cần thiết cho phép khởi tạo, nâng cấp cơ sở dữ liệu.
- Tạo đối tượng để truy cập cơ sở dữ liệu (Read và Write).



2. Xây dựng CSDL với SQLite



❑ Đổi tượng SQLiteDatabase:

- Tạo đối tượng để truy cập cơ sở dữ liệu (Read và Write).
- SQLiteDatabase cung cấp phương thức insert(), update(), delete(), hoặc execSQL() cho phép thực hiện truy xuất dữ liệu.



2. Xây dựng CSDL với SQLite



❑ Tạo Database:

```
@Override
public void onCreate(SQLiteDatabase _db) {
    _db.execSQL(DATABASE_CREATE);
}
```

❑ Mở Database:

```
SQLiteDatabase db;
try {
    db = dbHelper.getWritableDatabase();
}
catch (SQLException ex){
    db = dbHelper.getReadableDatabase();
}
```



Thảo luận



cuu duong than cong . com

Lập trình Android (2014) – Bài 2. SQLite

10

cuu duong than cong . com



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

Lập trình Android

Bài 3. *Truy vấn và sắp xếp dữ liệu*

Ngành Mạng & Thiết bị di động

cuu duong than cong . com
2014



5014



cuu duong than cong . com

Nội dung



1. Cursor

- Khái niệm
- Dữ liệu Cursor từ SQLite
- Các hàm chức năng CSDL
- Các phương thức điều khiển



1.1 Khái niệm



□ Cursor:

- Đối tượng dữ liệu được trả về khi thực hiện truy vấn dữ liệu trong cơ sở dữ liệu của SQLite hoặc Content Provider.
- Thể hiện dữ liệu ở dạng bảng quan hệ :
 - Cột: thể hiện trường dữ liệu (hoặc thuộc tính dữ liệu)
 - Dòng: một thể hiện dữ liệu

word	app id	frequency	locale	_ID
mapreduce	user1	100	en_US	1
precompiler	user14	200	fr_FR	2
applet	user2	225	fr_CA	3
const	user1	255	pt_BR	4
int	user5	100	en_UK	5



1.2 Dữ liệu Cursor từ SQLite



❑ Truy vấn cơ sở dữ liệu SQLite:

- Sử dụng phương thức rawQuery():

```
    Cursor cursor = getReadableDatabase().rawQuery
        ("select * from todo where _id = ?", new String[] { id });
```

- Sử dụng phương thức query():

```
    Cursor cursor = database.query
        (String table, String[] columns,
         String selection, String[] selectionArgs,
         String groupBy, String having, String orderBy);
```



1.2 Dữ liệu Cursor từ SQLite



❑ Tham số cho câu truy vấn:

- String TABLE: tên của bảng cần truy vấn
- String[] columns: danh sách các cột sẽ trả về dữ liệu
- String selection: chứa các điều kiện truy vấn
- String[] selectionArgs: danh sách các tham số phụ cho câu điều kiện
- String[] groupBy: gom nhóm các cột kết quả
- String[] having : bộ lọc theo điều kiện
- String[] orderBy : sắp xếp theo mảng cột được chỉ định





1.2 Dữ liệu Cursor từ SQLite

❑ Ví dụ truy vấn:

```
// Return all rows for columns one and three, no duplicates
String[] result_columns = new String[] {KEY_ID, KEY_COL1, KEY_COL3};

Cursor allRows = myDatabase.query(true, DATABASE_TABLE, result_columns,
                                    null, null, null, null, null, null);

// Return all columns for rows where column 3 equals a set value
// and the rows are ordered by column 5.
String where = KEY_COL3 + "=" + requiredValue;
String order = KEY_COL5;
Cursor myResult = myDatabase.query(DATABASE_TABLE, null, where,
                                      null, null, null, order);
```



1.3 Các hàm chức năng CSDL

❑ Thêm dữ liệu:

```
// Create a new row of values to insert.
ContentValues newValues = new ContentValues();

// Assign values for each row.
newValues.put(COLUMN_NAME, newValue);
[ ... Repeat for each column ... ]

// Insert the row into your table
myDatabase.insert(DATABASE_TABLE, null, newValues);
```



1.3 Các hàm chức năng CSDL



❑ Cập nhật dữ liệu:

```
// Define the updated row content.
ContentValues updatedValues = new ContentValues();

// Assign values for each row.
newValues.put(COLUMN_NAME, newValue);
[ ... Repeat for each column ... ]

String where = KEY_ID + "=" + rowId;

// Update the row with the specified index with the new values.
myDatabase.update(DATABASE_TABLE, newValues, where, null);
```



1.3 Các hàm chức năng CSDL



❑ Xoá dữ liệu:

- Sử dụng câu truy vấn delete truyền vào tên bảng và chỉ số của hàng cần xoá trong mệnh đề where.
- Nếu mệnh đề where không có sẽ thực hiện xoá tất cả các dòng.

❑ Ví dụ: cuu duong than cong . com

- myDatabase.delete
(DATABASE_TABLE, KEY_ID + " = " + rowID, null);



1.4 Các phương thức điều khiển



❑ Một số các phương thức để điều khiển Cursor thao tác trên dữ liệu:

- moveToFirst
- moveToNext
- moveToPrevious
- moveToPosition
- getCount
- getColumnIndexOrThrow
- getColumnName
- getColumnNames
- getPosition



Thảo luận





Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

Lập trình Android

Bài 4. Menu

Ngành Mạng & Thiết bị di động

cuu duong than cong . com
2014



5014



cuu duong than cong . com

Nội dung



1. Menu

- Options Menu
- Context Menu
- Popup Menu
- Checkable Menu



cuu duong than cong . com

2

1. Menu



Dạng Widget chứa các chức năng phụ hoặc các tùy chỉnh dành riêng cho từng ứng dụng. Bao gồm 4 dạng cơ bản:

- Option menu
- Context menu
- PopUp Menu



cuu duong than cong . com

3

1.1 Option Menu

❑ Option Menu:

- Là Menu chính trong ứng dụng chứa các thao tác cơ bản cho một ứng dụng được gọi khi người dùng nhấn phím Menu.
- Từ phiên bản Android 2.3 trở xuống,
- thao tác gọi Menu được thực hiện
- bằng phím Menu trên thiết bị.



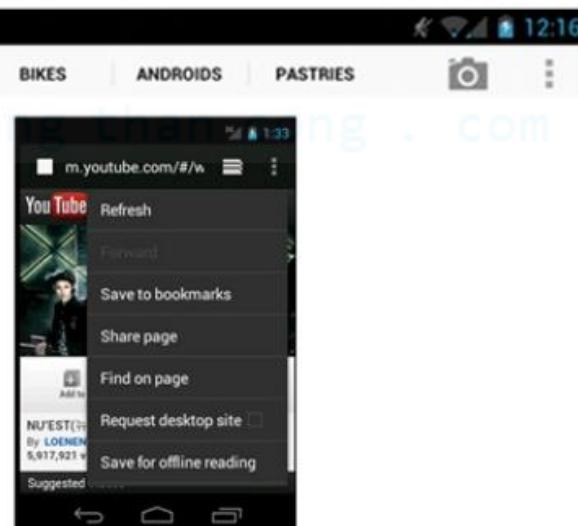
Lập trình Android (2014) – Bài 4. Menu

4

1.1 Option Menu

❑ Option Menu:

- Từ phiên bản Android 3.0 trở đi Option Menu được tích hợp vào trong thanh Action Bar.



Lập trình Android (2014) – Bài 4. Menu

5

1.1 Option Menu



□ Tạo Option Menu từ XML:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/new_game"
          android:icon="@drawable/ic_new_game"
          android:title="@string/new_game"
          android:showAsAction="ifRoom"/>
    <item android:id="@+id/help"
          android:icon="@drawable/ic_help"
          android:title="@string/help" />
</menu>
```



1.1 Option Menu



□ Tham chiếu trong Java code qua hàm onCreateOptionsMenu():

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // TODO Auto-generated method stub
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.game_menu, menu);
    return true;
}
```





1.1 Option Menu

□ Xử lý sự kiện chọn trong OptionMenu:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle item selection
    switch (item.getItemId()) {
        case R.id.new_game:
            newGame();
            return true;
        case R.id.help:
            showHelp();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```



1.2 Context Menu

□ Context Menu:

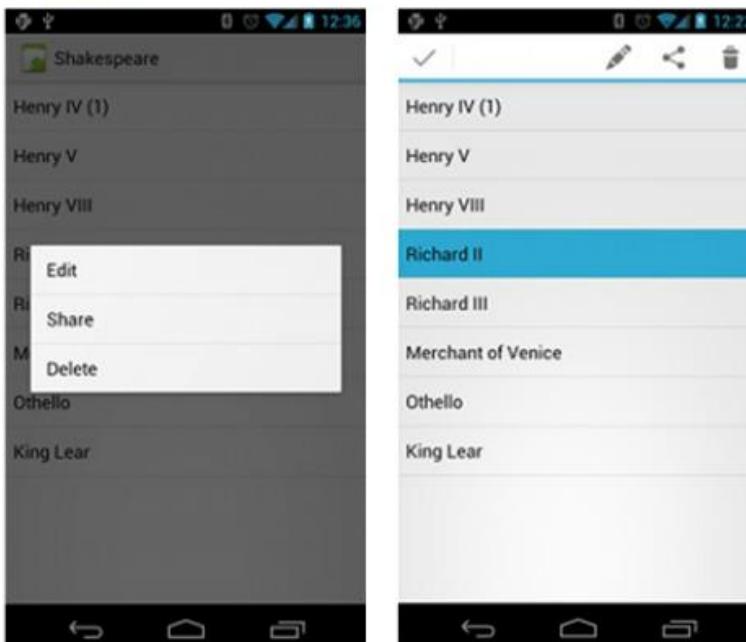
- Dạng menu xuất hiện khi người dùng tương tác với các Item trên ViewGroup, thường là ListView hoặc GridView.
- Bao gồm 2 dạng:
 - Floating Context Menu: dạng menu hiển thị khi người dùng nhấn và giữ một item trên ViewGroup (giống như Dialog).
 - Contextual Action Mode (API level 11): một thanh công cụ hiển thị phía trên ứng dụng cho phép người thực hiện nhiều thao tác khác nhau trên Item được lựa chọn, hoặc thực hiện một thao tác trên nhiều Item nếu ứng dụng có hỗ trợ.



1.2 Context Menu



❑ Context Menu:



1.2 Context Menu



❑ Khởi tạo Floating Context Menu

- Đăng ký đối tượng View sẽ sử dụng bằng phương thức `registerForContextMenu(View)`
- Thực hiện override phương thức `onCreateContextMenu()`

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                               ContextMenuItemInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.context_menu, menu);
}
```



1.2 Context Menu



- Xử lý sự kiện trên Floating Context Menu bằng cách override phương thức **onContextItemSelected()**

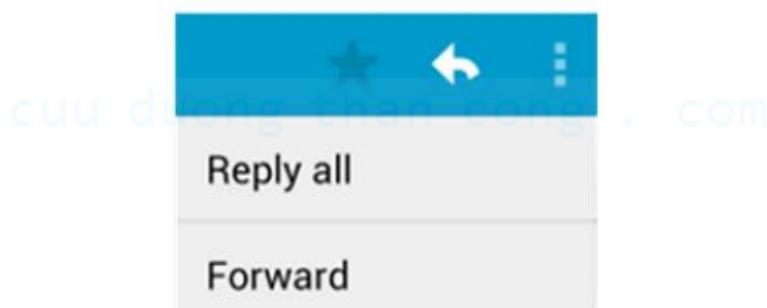
```
@Override
public boolean onContextItemSelected(MenuItem item) {
    AdapterContextMenuInfo info = (AdapterContextMenuInfo) item.getMenuInfo();
    switch (item.getItemId()) {
        case R.id.edit:
            editNote(info.id);
            return true;
        case R.id.delete:
            deleteNote(info.id);
            return true;
        default:
            return super.onContextItemSelected(item);
    }
}
```



1.3 Popup Menu



- Dạng menu hiển thị khi người dùng nhấn và giữ lâu trên một đối tượng.



1.3 Popup Menu



❑ Thực hiện khởi tạo Popup Menu cho một đối tượng:

- Ví dụ gọi phương thức onClick từ một Button

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_overflow_holo_da
rk"
    android:contentDescription
    ="@string/descr_overflow_button"
    android:onClick="showPopup" />
```



1.3 Popup Menu



❑ Thực hiện khởi tạo Popup Menu cho một đối tượng:

- Xử lý phương thức showPopup

```
public void showPopup(View v) {
    PopupMenu popup = new PopupMenu(this, v);
    MenuInflater inflater = cong . com
    popup.getMenuInflater();
    inflater.inflate(R.menu.actions,
    popup.getMenu());      popup.show();
}
```



1.3 Popup Menu

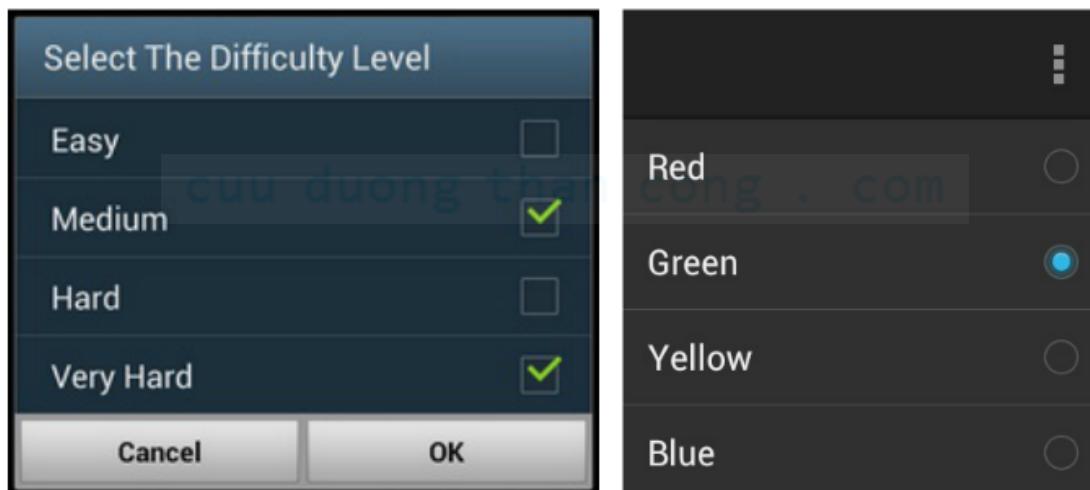
❑ Xử lý sự kiện khi chọn trên Popup Menu:

```
@Override public boolean onMenuItemClick(MenuItem item) { switch (item.getItemId()) { case R.id.archive: archive(item); return true; case R.id.delete: delete(item); return true; default: return false; } }
```



1.4 Checkable Menu

❑ Các dạng Menu hỗ trợ người đưa các lựa chọn:



1.4 Checkable Menu



❑ Khai báo chế độ chọn cho Item trong XML:

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res
    /android">
    <group android:checkableBehavior="single">
        <item android:id="@+id/red"
            android:title="@string/red" />
        <item
            android:id="@+id/blue"
            android:title="@string/blue" />
    </group>
</menu>
```



1.4 Checkable Menu



❑ Xử lý sự kiện chọn của Menu:

```
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
    switch (item.getItemId()) {
        case R.id.vibrate:
        case R.id.dont_vibrate:
            if (item.isChecked())
                item.setChecked(false);
            else item.setChecked(true);
            return true;
        default:
            return
    super.onOptionsItemSelected(item);
    }
}
```



Thảo luận



cuu duong than cong . com

Lập trình Android (2014) – Bài 4. Menu

20

cuu duong than cong . com



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

Lập trình Android

Bài 5. Action Bar

Ngành Mạng & Thiết bị di động

cuu duong than cong . com

2014



2014



cuu duong than cong . com

Nội dung



1. Action Bar

- Tổng quan thuộc tính
- ActionView trên ActionBar
- ActionProvider
- Chế độ điều hướng



1.1 Tổng quan



□ ActionBar:

- Điều hướng giao diện ứng dụng.
- Hiển thị các thao tác trên toàn bộ hệ thống ứng dụng hoặc thao tác tại màn hình hiển thị (tìm kiếm, chỉnh sửa, xóa...).
- Thao tác chuyển đến các giao diện khác nhau trong cùng một màn hình hiển thị (tabhost, list navigation...).



1.1 Tổng quan



- **ActionBar: được cung cấp sử dụng từ phiên bản Android 3.0**
 - Bao gồm hai phiên bản ở thời điểm hiện tại:
 - Hỗ trợ các phiên bản thấp hơn API 11:
 - android.support.v7.app.ActionBar
 - Khai báo sử dụng:
 - Activity kế thừa từ lớp ActionBarActivity
 - Sử dụng hoặc kế thừa Theme.AppCompat trong Style
 - Ví dụ:
`<activity android:theme="@style/Theme.AppCompat.Light" ... >`
 - Dành cho các phiên bản từ API 11 trở lên:
 - android.app.ActionBar
 - Khai báo sử dụng:
 - minSDKversion >= 11 trong AndroidManifest.xml



1.1 Tổng quan



- **ActionBar:**
 - Khai báo sử dụng trong Java Code:
 - android.support.v7.app.ActionBar
`ActionBar actionBar = getSupportActionBar();`
 - android.app.ActionBar
`ActionBar actionBar = getSupportActionBar();`
 - Hiển thị - giấu ActionBar
 - `actionBar.show(); // mặc định`
 - `actionBar.hide();`
- **Lưu ý: Ứng dụng sử dụng việc hiển thị - giấu ActionBar liên tục có thể thay thế bằng chế độ Overlay trên Android 4.3 & 4.4**





1.1 Tổng quan

□ ActionBar:

- SplitActionBar: cơ chế cho phép hiển thị ActionBar ở dạng chia đôi khi có nhiều item.



▪ Sử dụng:

Khai báo thuộc tính uiOptions="splitActionBarWhenNarrow" trong thẻ `<activity>`



1.1 Tổng quan

□ ActionBar:

- Ví dụ khai báo cho SplitActionBar cho các phiên bản cũ hơn:

▪ Cần khai báo thêm thẻ `<meta-data>`

```
<manifest ...>
    <activity uiOptions="splitActionBarWhenNarrow" ... >
        <meta-data android:name="android.support.UI_OPTIONS"
                  android:value="splitActionBarWhenNarrow" />
    </activity>
</manifest>
```



1.1 Tổng quan



□ ActionBar:

- Navigation Up Icon: chế độ hiển thị và tương tác với biểu tượng ActionBar cho phép người dùng có thể quay về màn hình trước đó ứng dụng (tùy thuộc ngữ cảnh).



- Sử dụng: khai báo thông qua phương thức setDisplayHomeAsUpEnabled()

- Ví dụ:

```
ActionBar actionBar = getSupportActionBar();
actionBarsetDisplayHomeAsUpEnabled(true);
```



1.1 Tổng quan



□ ActionBar:

- Navigation Up Icon:



- Khai báo sử dụng trong AndroidManifest.xml

- Ví dụ:

```
<activity
    android:name="com.htsi.t3h.SecondActivity"
    android:label="@string/title_activity_display_message" >
```

```
    android:parentActivityName="com.htsi.t3h.MainActivity">
```

```
<!-- Hỗ trợ khai báo cho các phiên bản cũ hơn-->
```

```
<meta-data
    android:name="android.support.PARENT_ACTIVITY"
    android:value="com.htsi.t3h.MainActivity" />
```

```
</activity>
```



1.2 ActionView trênActionBar



- ActionView: dạng điều khiển hiển thị trên ActionBar cho phép thực hiện các chuỗi thao tác trên cùng một điều khiển để thay đổi dữ liệu.
 - Khai báo sử dụng thông qua hai thuộc tính:
 - actionLayout
 - actionViewClass



1.2 ActionView trênActionBar



□ ActionView:

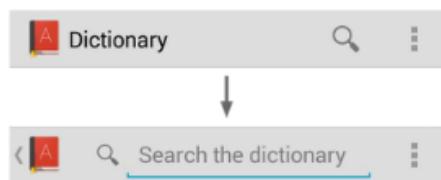
- SearchView

- Khai báo sử dụng trong XML

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto" >
```

```
<item android:id="@+id/action_search"
      android:title="@string/action_search"
      android:icon="@drawable/ic_action_search"
      app:showAsAction="ifRoom|collapseActionView"
      app:actionViewClass="android.support.v7.widget.SearchView" />
```

```
</menu>
```



1.2 ActionView trênActionBar



□ ActionView:

- SearchView

- Tham chiếu đối tượng trong JavaCode: cần thực hiện trong phương thức khởi tạo Menu – onCreateOptionsMenu.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
```

```
    getMenuInflater().inflate(R.menu.main_activity_actions, menu);
```

```
    MenuItem searchItem = menu.findItem(R.id.action_search);
```

```
    SearchView searchView = (SearchView)
        MenuItemCompat.getActionView(searchItem);
```

```
    // Tùy chỉnh đối tượng cũng như các hàm lắng nghe sự kiện
```

```
    ...
    return super.onCreateOptionsMenu(menu);
}
```



1.2 ActionView trênActionBar



□ ActionView:

- SearchView

- Lắng nghe sự kiện Collapse/Expand của SearchView:

```
    MenuItemCompat.setOnActionExpandListener(menuItem, new
OnActionExpandListener() {
```

```
        @Override
```

```
        public boolean onMenuItemActionCollapse(MenuItem item) {
```

```
            // Collapsed
```

```
            return true;
```

```
        }
```

```
        @Override
```

```
        public boolean onMenuItemActionExpand(MenuItem item) {
```

```
            // Expanded
```

```
            return true;
```

```
        }
```

```
    };
```

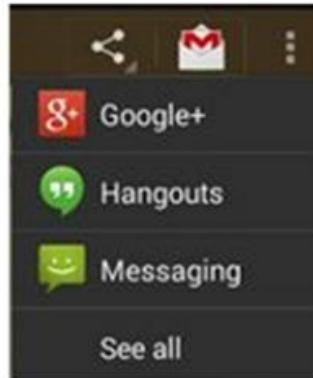
```
}
```





1.3 ActionProvider

- **ActionProvider:** dạng điều khiển giống ActionView, tuy nhiên ActionProvider chỉ bao gồm hành động xử lý đến từ ứng dụng bên ngoài, có thể xây dựng thông qua việc kế thừa lớp ActionProvider.
 - ShareActionProvider: lớp kế thừa được Android xây dựng sẵn, bao gồm các tiện ích đến các ứng dụng có khả năng chia sẻ.



1.3 ActionProvider

- **ActionProvider**
 - ShareActionProvider:
 - Khai báo sử dụng trong XML

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto" >

    <item android:id="@+id/action_share"
          android:title="@string/share"
          app:showAsAction="ifRoom"
          app:actionProviderClass=
              "android.support.v7.widget.ShareActionProvider" />
</menu>
```



1.3 ActionProvider



□ ActionProvider

- ShareActionProvider:

- Tham chiếu đối tượng trong JavaCode: cần thực hiện trong phương thức khởi tạo Menu – onCreateOptionsMenu.

@Override

```
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main_activity_actions, menu);
```

// Set up ShareActionProvider's default share intent

```
MenuItem shareItem = menu.findItem(R.id.action_share);
```

```
ShareActionProvider mShareActionProvider = (ShareActionProvider)
    MenuItemCompat.getActionProvider(shareItem);
```

```
mShareActionProvider.setShareIntent(getDefaultIntent());
```

```
return super.onCreateOptionsMenu(menu);
```

```
}
```



1.4 Chế độ điều hướng



□ NavigationMode: cho phép hiển thị dữ liệu màn hình ở hai chế độ

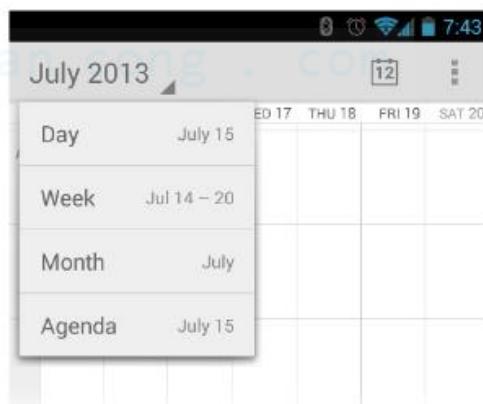
- Tab:

- Dữ liệu màn hình bao gồm nhiều trang, mỗi trang được điều hướng hiển thị theo tab của ActionBar.



- List:

- Dữ liệu màn hình có thể bao gồm nhiều trang hoặc chế độ hiển thị, mỗi chế độ được chọn lựa từ Spinner thiết lập trên ActionBar.





1.4 Chế độ điều hướng

- **NavigationMode:** thực hiện khai báo chế độ điều hướng thông qua phương thức `setNavigationMode`.

- Tab:

- Khai báo:

```
actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
```

- Xây dựng Tab:

- Gọi phương thức `newTab()`: đối tượng trả về - Tab

- Thiết lập bộ lắng nghe - `setTabListener`
 - Biểu tượng - `setIcon`
 - Tiêu đề cho từng Tab - `setText`

- Gắn Tab vào ActionBar thông qua phương thức `addTab`, tham số truyền vào là một đối tượng Tab.



1.4 Chế độ điều hướng

- **NavigationMode:**

- Tab:

- Ví dụ khai báo sử dụng:

```
Tab tab = actionBar.newTab()  
        .setText(R.string.artist)  
        .setTabListener(new TabListener<ArtistFragment>(  
            this, "artist", ArtistFragment.class));  
actionBar.addTab(tab);
```

```
tab = actionBar.newTab()  
        .setText(R.string.album)  
        .setTabListener(new TabListener<AlbumFragment>(  
            this, "album", AlbumFragment.class));  
actionBar.addTab(tab);
```



1.4 Chế độ điều hướng



- **NavigationMode:** thực hiện khai báo chế độ điều hướng thông qua phương thức `setNavigationMode`.

- List:

- Khai báo:

```
actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_LIST);
```

- Xây dựng List:

- Tạo đối tượng `SpinnerAdapter`

- Khai báo bộ lắng nghe sự kiện trên List - `OnNavigationListener`

- Gọi phương thức `setListNavigationCallbacks`, tham số truyền vào là một adapter và bộ lắng nghe sự kiện.



1.4 Chế độ điều hướng



- **NavigationMode:**

- Ví dụ :

- Ví dụ khai báo sử dụng :

```
// Khai báo Adapter
```

```
SpinnerAdapter mSpinnerAdapter = ArrayAdapter.createFromResource
    (this, R.array.action_list,
     android.R.layout.simple_spinner_dropdown_item);
```

```
// OnNavigationListner
```

```
OnNavigationListner mNavigationCallback= new OnNavigationListener()
{...}
```

```
// Thiết lập Adapter và lắng nghe sự kiện
```

```
actionBar.setListNavigationCallbacks(mSpinnerAdapter,
    mNavigationCallback);
```



Thảo luận



cuu duong than cong . com

Lập trình Android (2014) – Bài 5. Action Bar

22

cuu duong than cong . com



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

Lập trình Android

Bài 6. *Animation*

Ngành Mạng & Thiết bị di động

cuu duong than cong . com
2014



5014



cuu duong than cong . com

Nội dung



1. Animation trong tài nguyên XML

- Property Animation
- View Animation
- Drawable Animation



1. Animation trong tài nguyên XML



Property Animation

- Cho phép thiết lập các thay đổi thuộc tính của đối tượng theo thời gian, bất kể đối tượng có thể hiện trên màn hình hay không.

View Animation

- Thực hiện một chuỗi các thao tác chuyển đổi đơn giản trên một đối tượng **View** duy nhất.

Drawable Animation:

- Thực hiện các chuyển đổi của các đối tượng Drawable theo một trật tự được định trong một khoảng thời gian.

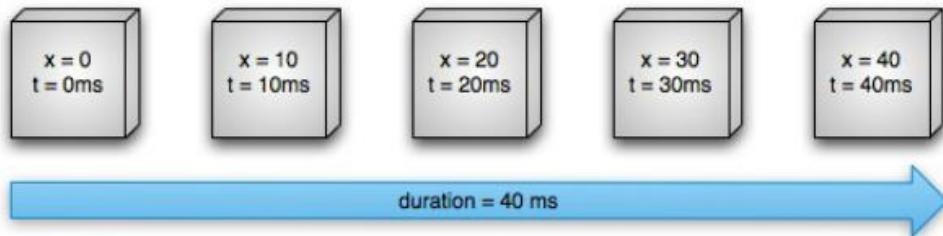


1.1 Property Animation



- ❑ Thư mục tài nguyên:
- res/animator/filename.xml

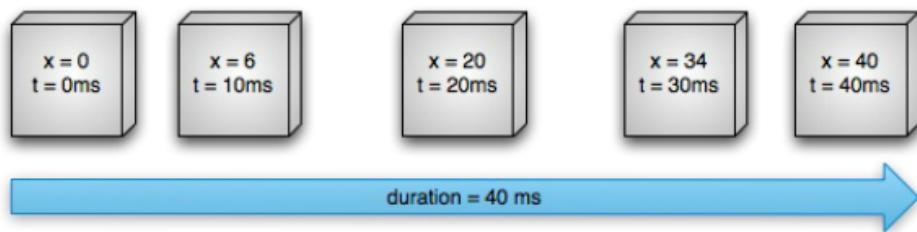
- ❑ Cách thức hoạt động:



1.1 Property Animation



- ❑ Các thuộc tính có thể tùy chỉnh trong Property Animation:
- Duration
- Time interpolation
- Repeat count
- Animation Sets
- Frame refresh delay



1.1 Property Animation



❑ Khai báo các thuộc tính trong XML:

- ValueAnimator - <animator>

```
<animator
    android:duration="int"
    android:valueFrom="float | int | color"
    android:valueTo="float | int | color"
    android:startOffset="int"
    android:repeatCount="int"
    android:repeatMode=["repeat" | "reverse"]
    android:valueType=["intType" | "floatType"]/>
```



1.1 Property Animation



❑ Khai báo các thuộc tính trong XML:

- ObjectAnimator - <objectAnimator>

```
<objectAnimator
    android:propertyName="string"
    android:duration="int"
    android:valueFrom="float | int | color"
    android:valueTo="float | int | color"
    android:startOffset="int"
    android:repeatCount="int"
    android:repeatMode=["repeat" | "reverse"]
    android:valueType=["intType" | "floatType"]/>
```



1.1 Property Animation



❑ Khai báo các thuộc tính trong XML:

- AnimatorSet - <set>

```
<set  
    android:ordering=["together" | "sequentially"]>  
    <set>  
        ...  
    </set>  
</set>
```



1.1 Property Animation



❑ Ví dụ khai báo trong PropertyAnimation:

- exam_object_animator.xml:

```
<objectAnimator  
    android:propertyName="x"  
    android:duration="2000"  
    android:valueTo="500"  
    android:repeatMode="restart"  
    android:repeatCount="infinite">  
</objectAnimator>
```



1.1 Property Animation



❑ Ví dụ khai báo trong PropertyAnimation:

- exam_animator_set.xml:

```
<set android:ordering="sequentially">
    <objectAnimator
        android:propertyName="x"
        android:duration="2000" android:repeatMode="reverse"
        android:valueTo="400" android:repeatCount="infinite"
        android:valueType="intType"/>
    <objectAnimator
        android:propertyName="y"
        android:duration="2000" android:repeatMode="reverse"
        android:valueTo="400"
        android:repeatCount="infinite"           android:valueType="intType"/>
</set>
```



1.1 Property Animation



❑ Tham chiếu và sử dụng trong Java code:

- Thực hiện tham chiếu thông qua phương thức **loadAnimator** trong lớp **AnimatorInflater**.
- Gọi phương thức **setTarget**, tham số truyền vào là đối tượng cần thực hiện Animator.
- Gọi phương thức **start()** từ đối tượng Animator.
- Cần thực ép kiểu đúng đối tượng tương ứng trong XML.
- Ví dụ tham chiếu ScaleAnimation:

ObjectAnimator **objectAnimator** =

```
(ObjectAnimator ) AnimatorInflater.loadAnimator(this,
    R.animator.exam_object_animator);
```

```
objectAnimator.setTarget(imageView);
```

```
objectAnimator.start();
```



1.1 Property Animation



❑ Tham chiếu và sử dụng trong Java code:

- Ví dụ tham chiếu AnimatorSet:

```
AnimatorSet animatorSet =
    (AnimatorSet) AnimatorInflater.loadAnimator(this,
        R.animator.my_animator_set);
animatorSet.setTarget(imageView);
animatorSet.start();
```



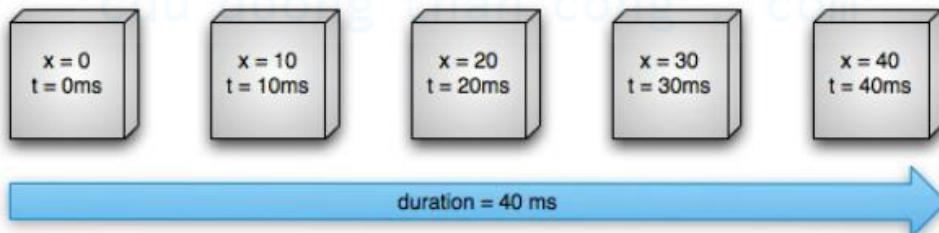
1.2 View Animation



❑ Thư mục tài nguyên:

- res/anim/filename.xml

❑ Cách thức hoạt động:





1.2 View Animation

❑ Các chuyển hoạt trong View Animation:

- AlphaAnimation - <alpha>

```
<alpha>  
    android:fromAlpha="float"  
    android:toAlpha="float" />
```

- ScaleAnimation - <scale>

```
<scale>  
    android:fromXScale="float"  
    android:toXScale="float"  
    android:fromYScale="float"  
    android:toYScale="float"  
    android:pivotX="float"  
    android:pivotY="float" />
```



1.2 View Animation

❑ Các chuyển hoạt trong View Animation:

- TranslateAnimation - <translate>

```
<translate>  
    android:fromXDelta="float"  
    android:toXDelta="float"  
    android:fromYDelta="float"  
    android:toYDelta="float" /> Scale
```

- RotateAnimation - <rotate>

```
<rotate>  
    android:fromDegrees="float"  
    android:toDegrees="float"  
    android:pivotX="float"  
    android:pivotY="float" />
```



1.2 View Animation



- ❑ Tổng hợp các chuyển hoạt trong một chuyển hoạt duy nhất của View Animation:

- AnimationSet - <set>

```
<set xmlns:android="http://schemas.android.com/apk/res/
    android"
    android:interpolator="@[package:]anim/interpolator_resource"
    android:shareInterpolator=["true" | "false"] >
```

```
<set>
```

```
...
```

```
</set>
```

```
</set>
```



1.2 View Animation



- ❑ Ví dụ khai báo ScaleAnimation:

- exam_scale_anim.xml:

```
<scale xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromXScale="1"
    android:toXScale="2"
    android:fromYScale="1"
    android:toYScale="2"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="2000"
    android:fillAfter="true"
    android:repeatCount="infinite"
    android:repeatMode="reverse"
    >
```

```
</scale>
```





1.2 View Animation

❑ Ví dụ khai báo AnimationSet:

- exam_anim_set.xml:

```
<set xmlns:android="http://schemas.android.com/apk/res/android"  
      android:duration="1600" android:repeatCount="infinite">  
    <scale  
        android:fromXScale="1"  
        android:fromYScale="1"  
        android:toXScale="2"  
        android:toYScale="2" />  
    <rotate  
        android:fromDegrees="0"  
        android:toDegrees="360" />  
</set>
```



1.2 View Animation

❑ Tham chiếu và sử dụng trong Java code:

- Thực hiện tham chiếu thông qua phương thức **loadAnimation** trong lớp **AnimationUtils**.
- Gọi phương thức **startAnimation** từ đối tượng View cần thực hiện chuyển hoạt, tham số truyền vào là đối tượng Animation.
- Cần thực ép kiểu đúng đối tượng tương ứng trong XML.
- Ví dụ tham chiếu ScaleAnimation:

ScaleAnimation **scaleAnimation** =

```
(ScaleAnimation) AnimationUtils.loadAnimation(this,  
                                              R.anim.exam_scale_anim);
```

```
imageView.startAnimation(scaleAnimation);
```





1.2 View Animation

❑ Tham chiếu và sử dụng trong Java code:

- Ví dụ tham chiếu AnimationSet:

```
AnimationSet animationSet =  
    (AnimationSet) AnimationUtils.loadAnimation (this,  
        R.anim.exam_anim_set);  
  
imageView.startAnimation(animationSet);
```



1.3 Drawable Animation

❑ Thư mục tài nguyên:

- res/drawable/filename.xml

❑ Cách thức hoạt động:



1.3 Drawable Animation



❑ Khai báo các thuộc tính trong XML:

- AnimationDrawable - <animation-list>

```
<animation-list
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot=["true" | "false"] >
    <item  android:drawable="@[package:]drawable drawable_resource_name"
        android:duration="integer" />
</animation-list>
```



1.3 Drawable Animation



❑ Ví dụ khai báo Drawable Animation:

- exam_drawable_anim.xml:

```
<animation-list
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">
    <item android:drawable="@drawable/plane01"
        android:duration="200" />
    <item android:drawable="@drawable/plane02"
        android:duration="200" />
    <item android:drawable="@drawable/plane01"
        android:duration="200" />
</animation-list>
```



1.3 Drawable Animation



❑ Tham chiếu và sử dụng trong Java code:

- Thực hiện truy xuất đối tượng **AnimationDrawable** từ đối tượng View đang hiển thị Drawable, bằng phương thức **getBackgroundResource()**.
- Gọi phương thức **start()** từ đối tượng **AnimationDrawable**.
- Ví dụ tham chiếu **AnimationDrawable**:

```
ImageView planeView = (ImageView) findViewById(R.id.plane);
planeView.setBackgroundResource
        (R.drawable.exam_drawable_anim);
AnimationDrawable plane = (AnimationDrawable)
        planeView.getBackground();
plane.start();
```



Thảo luận





Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

Lập trình Android

Bài 7. CursorLoader

Ngành Mạng & Thiết bị di động

cuu duong than cong . com

2014



2014



cuu duong than cong . com

Nội dung



1. CursorLoader

- Khái niệm
- Xây dựng truy vấn CursorLoader

2. Sử dụng CursorLoader trong CursorAdapter



1.1 Khái niệm



❑ CursorLoader:

- Lớp kế thừa AsyncTaskLoader cho phép khởi tạo các câu truy vấn bắt đồng bộ, dựa trên cơ sở dữ liệu của ContentProvider.
- Thực hiện truy vấn và trả dữ liệu cho FragmentActivity và Activity ở chế độ ngầm.
- Các đối tượng CursorLoader được quản lý thông qua đối tượng LoaderManager tồn tại trong mỗi FragmentActivity và Activity.
- Dữ liệu trả về là một đối tượng Cursor.



1.1 Khái niệm



❑ CursorLoader:

- Bao gồm hai lớp riêng biệt được định nghĩa hỗ trợ cho các phiên bản cũ hơn của hệ điều hành.
 - `android.content.CursorLoader`
 - Khai báo cho các phiên bản API 11 trở lên
 - `android.support.v4.content.CursorLoader`
 - Khai báo hỗ trợ cho các phiên bản API 4 trở lên



1.2 Xây dựng truy vấn CursorLoader



❑ CursorLoader:

- Phương thức khởi tạo: đối tượng CursorLoader được khởi tạo thông qua LoaderManager.

▪ `initLoader(int id, Bundle args, LoaderCallBack<D> callBacks)`

Trong đó:

- Id: mỗi CursorLoad được khởi tạo kèm id để quản lý
- Args: tham số hỗ trợ cho câu truy vấn
- callBacks: lớp kế thừa LoaderCallBack cho phép thực hiện khởi tạo câu truy vấn và nhận kết quả trả về.

- Ví dụ khởi tạo Loader:

`getLoaderManager.initLoader(0, null, this);`



1.2 Xây dựng truy vấn CursorLoader



❑ CursorLoader:

- LoaderManger: đối tượng quản lý CursorLoader.
 - android.app.LoaderManager
 - getLoaderManager()
 - android.support.v4.app.LoaderManger
 - getSupportLoaderManager()



1.2 Xây dựng truy vấn CursorLoader



❑ CursorLoader:

- LoaderManger.LoaderCallBack<D>: lớp trừu tượng cho phép thực hiện khai báo các hàm khởi tạo câu truy vấn theo kiểu dữ liệu D.

- Ví dụ: khai báo lớp khởi tạo truy vấn với kiểu dữ liệu Cursor.

```
public class MyFragmentActivity extends FragmentActivity implements
    LoaderManager.LoaderCallbacks<Cursor>
{
    ...
}
```



1.2 Xây dựng truy vấn CursorLoader



❑ CursorLoader:

- LoaderManger.LoaderCallBack<D>: cần thực hiện override các phương thức hỗ trợ xây dựng câu truy vấn .
 - onCreateLoader
 - onLoadFinish
 - onLoadReset



1.2 Xây dựng truy vấn CursorLoader



❑ CursorLoader:

- LoaderManger.LoaderCallBack<D>

- Ví dụ khởi tạo câu truy vấn.

```
public Loader<Cursor> onCreateLoader(int id, Bundle args) {
    Uri baseUri = Contacts.CONTENT_URI;
    String select = "(" + Contacts.DISPLAY_NAME + " NOTNULL) AND (" +
        + Contacts.HAS_PHONE_NUMBER + "=1) AND (" +
        + Contacts.DISPLAY_NAME + " != " ")";
    return new CursorLoader(getActivity(), baseUri,
        CONTACTS_SUMMARY_PROJECTION, select, null,
        Contacts.DISPLAY_NAME + " COLLATE LOCALIZED ASC");
}
```



1.2 Xây dựng truy vấn CursorLoader



❑ CursorLoader:

- LoaderManger.LoaderCallBack<D>

- Thực hiện cập nhật dữ liệu khi có sự thay đổi.

```
public void onLoadFinished(Loader<Cursor> loader, Cursor data) {  
  
    mAdapter.swapCursor(data);  
}
```

- Thực hiện xoá tham chiếu dữ liệu cũ.

```
public void onLoaderReset(Loader<Cursor> loader, Cursor data) {  
  
    mAdapter.swapCursor(null);  
}
```



Nội dung



1. CursorLoader

2. Sử dụng CursorLoader kết hợp CursorAdapter



2. CursorLoader kết hợp CursorAdapter



❑ CursorAdapter:

- Dạng adapter cho phép thao tác dữ liệu trực tiếp từ Cursor.
- Cho phép kết nối trực tiếp và thay đổi dữ liệu AbsListView
- Dữ liệu truy vấn bắt buộc phải có trường “_id”
- Các biến cờ quan trọng:
 - FLAG_AUTO_REQUERY (Deprecated)
 - FLAG_REGISTER_CONTENT_OBSERVER
 - Tự động gọi phương thức onContentChanged khi có thông báo về thay đổi dữ liệu.



2. CursorLoader kết hợp CursorAdapter



❑ CursorAdapter: dạng adapter cho phép thao tác dữ liệu trực tiếp từ Cursor.

- Các phương thức quan trọng:
 - newView
 - khởi tạo giao diện từng item trong Adapter
 - bindView:
 - kết nối dữ liệu từ Cursor
 - runQueryOnBackgroundThread
 - Nhận chuỗi truy vấn và thực hiện truy vấn trên tiến trình ngầm,
 - setFilterQueryProvider
 - Khai báo bộ lọc Cursor, chỉ thực hiện truy vấn Cursor đầu vào đúng định dạng.



2. CursorLoader kết hợp CursorAdapter



❑ CursorAdapter.

- Ví dụ khai báo CursorAdapter:

```
public class MailAdapter extends CursorAdapter {
    ....
}

▪ Xây dựng giao diện Adapter:
@Override
public View newView(Context context, Cursor cursor, ViewGroup
viewGroup) {
    LayoutInflator inflater = LayoutInflator.from(context);
    View mailItemView = inflater.inflate(R.layout.email_item_layout,
null);
    TextView textSubject = (TextView) mailItemView
        .findViewById(R.id.textTitleMail);
    mailItemView.setTag(R.id.textTitleMail, textSubject);
    return mailItemView;
}
```



2. CursorLoader kết hợp CursorAdapter



❑ CursorAdapter.

- Ví dụ khai báo CursorAdapter:

- Kết nối dữ liệu vào giao diện Adapter:

@Override

```
public void bindView(View view, Context context, Cursor cursor) {
    TextView textSubject = (TextView)
        view.getTag(R.id.textTitleMail);

    String tempS =
        cursor.getString(cursor
            .getColumnIndex(MailData.COL_MAIL_SUBJECT));
    if (tempS.length() > 20) {
        textSubject.setText(tempS.substring(0, 20) + "...");
    } else {
        textSubject.setText(tempS);
    }
}
```





2. CursorLoader kết hợp CursorAdapter

❑ CursorAdapter.

- Ví dụ thay đổi dữ liệu CursorAdapter trong LoaderCallBack<Cursor>:

- Xử lý câu truy vấn:

```
@Override
```

```
public Loader<Cursor> onCreateLoader(int id, Bundle args) {
```

```
    switch (query) {
```

```
        case Mail.INBOX:
```

```
            Log.d("HTSI", "Query inbox mail");
```

```
            selection = MailData.COL_MAIL_TYPE + " = " +
```

```
Mail.INBOX;
```

```
            return new CursorLoader(context,
```

```
                MailDataContract.EMAIL_TABLE_CONTENTURI, null, selection,
```

```
                null, null);
```

```
}
```

```
}
```



2. CursorLoader kết hợp CursorAdapter

❑ CursorAdapter.

- Ví dụ thay đổi dữ liệu CursorAdapter trong LoaderCallBack<Cursor>:

- Xử lý câu truy vấn:

```
@Override
```

```
public void onLoadFinished(Loader<Cursor> arg0, Cursor cursor) {
```

```
    if (adapter == null) {
```

```
        adapter = new MailAdapter(context, cursor);
```

```
        listMail.setAdapter(adapter);
```

```
        return;
```

```
}
```

```
        adapter.changeCursor(cursor);
```

```
}
```



Thảo luận



cuu duong than cong . com

Lập trình Android (2014) – Bài 7. CursorLoader

18

cuu duong than cong . com



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

Lập trình Android

Bài 8. *AsynTask – Thread & Handler*

Ngành Mạng & Thiết bị di động

cuu duong than cong . com

2014



2014



cuu duong than cong . com

Nội dung



1. Thread và Handler

- Khái niệm
- Các phương thức quan trọng trong Thread
- HandleMessage
- Xây dựng tương tác cho Thread và Handler

2. AsyncTask



2.1 Khái niệm



Thread:

- Mỗi ứng dụng khi được khởi chạy sẽ được hoạt động trên tiến trình chính (mainThread).



Nội dung



1. Thread và Handler
2. AsyncTask
 - Khái niệm
 - Thuộc tính
 - Khởi tạo AsyncTask



2.1 Khái niệm



❑ AsyncTask:

- Lớp trừu tượng cho phép thực hiện khai báo các tiến trình xử lý ngầm và cập nhật giao diện một cách tự động.
- Bao gồm cơ chế tự động tính toán một cách bắt đồng bộ để đưa ra thời gian thực thi cho mỗi tiến trình xử lý.
- Thời gian xử lý được tính toán thông qua ba tham số:
 - Params: các thông số đầu vào
 - Progress: dữ liệu trong quá trình xử lý
 - Result: kết quả cuối cùng của tiến trình xử lý



2.2 Các phương thức quan trọng



❑ AsyncTask:

- Các phương thức cho phép nắm giữ dữ liệu ở nhiều giai đoạn khác nhau của tiến trình xử lý bao gồm 4 bước:
 - `onPreExecute`
 - Tiến trình tiền xử lý
 - `dolnBackground`
 - Tiến trình đang ở chế độ ngầm
 - `onProgressUpdate`
 - Tiến trình cập nhật dữ liệu
 - `onPostExecute`
 - Tiến trình cập nhật dữ liệu trên UI



2.2 Các phương thức quan trọng



❑ AsyncTask:

- Các qui tắc quan trọng khi xử lý trên AsyncTask:
 - **AsyncTask phải được tạo, gọi và thực thi trên tiến trình UI**
 - Tự động hóa từ phiên bản JellyBean
 - Không được gọi tường minh các phương thức xử lý `onPostExecute`, `dolnBackground`...
 - Mỗi AsyncTask chỉ được gọi một lần duy nhất.
 - Có thể thực hiện huỷ AsyncTask thông qua phương thức `onCancel(boolean)`
 - Khi đó phương thức `onCancel(Object)` sẽ được gọi sau phương thức `dolnBackground` thay vì `onPostExecute`.



2.3 Khởi tạo AsyncTask



❑ AsyncTask:

- Có thể tạo AsyncTask thông qua kế thừa lớp AsyncTask.
- Ví dụ tạo lớp AsyncTask và thực thi xử lý trong doInBackground

```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {  
    protected Long doInBackground(URL... urls) {  
        int count = urls.length;  
        long totalSize = 0;  
        for (int i = 0; i < count; i++) {  
            totalSize += Downloader.downloadFile(urls[i]);  
            publishProgress((int) ((i / (float) count) * 100));  
            if (isCancelled()) break;  
        }  
        return totalSize;  
    }  
}
```



2.3 Khởi tạo AsyncTask



❑ AsyncTask:

- Ví dụ cập nhật dữ liệu

```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {  
    ....  
    protected void onProgressUpdate(Integer... progress) {  
        setProgressPercent(progress[0]);  
    }  
  
    protected void onPostExecute(Long result) {  
        showDialog("Downloaded " + result + " bytes");  
    }  
}
```



Thảo luận



cuuduongthancong.com

Lập trình Android (2014) – Bài 8. AsyncTask – Threads & Handler

10

cuuduongthancong.com



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

Lập trình Android

Bài 9. *Broadcast Receiver & Service*

Ngành Mạng & Thiết bị di động

cuu duong than cong . com

2014



2014



cuu duong than cong . com



Nội dung

1. Broadcast Receiver

- Khái niệm
- Khai báo thuộc tính
- Đăng ký và sử dụng

2. Service



1.1 Khái niệm

- ❑ **Broadcast Receiver:** được sử dụng để nhận những Intent message được gửi từ sendBroadcast() hoặc từ hệ thống.
- ❑ **Để đăng ký đối tượng của Broadcast receiver, sử dụng 2 cách:**
 - Đăng ký động với phương thức
 - Context.registerReceiver()
 - Đăng ký trong Manifest.xml thông qua
 - <receiver/>



1.2 Khai báo thuộc tính



□ Đăng ký trong Manifest.xml

```
<receiver android:enabled=["true" | "false"]
          android:exported=["true" | "false"]
          android:icon="drawable resource"
          android:label="string resource"
          android:name="string"
          android:permission="string"
          android:process="string" >
    ...
</receiver>
```



1.2 Khai báo thuộc tính



□ Có 2 thành phần chính của Broadcast có thể nhận Intent

- Normal broadcasts: thực hiện gửi theo cơ chế không đồng bộ, các Receiver thực thi không theo thứ tự nhất định. Sử dụng phương thức `sendBroadcast()`.
- Ordered broadcasts: Vào mỗi thời điểm chỉ gửi đến một broadcast và sử dụng phương thức `sendOrderedBroadcast()`.



1.2 Khai báo thuộc tính



❑ Vòng đời của Broadcast Receiver

- Broadcast chỉ hoạt động trong thời gian onReceive() được gọi, và khi kết thúc hàm này, thì broadcast đã hoàn thành và không còn hoạt động nữa.

❑ Sự cung cấp quyền

- Chỉ Broadcast receiver được cung cấp quyền mới có thể nhận những intent gửi đi.



1.3 Đăng ký và sử dụng



❑ Tạo Broadcast Receiver:

```
public class EarthquakeReceiver extends BroadcastReceiver {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        //To do thing  
    }  
}
```



1.3 Đăng ký và sử dụng



❑ Khai báo và đăng ký Broadcast Receiver

```
IntentFilter filter = new IntentFilter(EarthquakeService.NEW_EARTHQUAKE_FOUND);
receiver = new EarthquakeReceiver();
registerReceiver(receiver, filter);
```

❑ Hủy đăng ký cho Broadcast

```
unregisterReceiver(receiver);
```



1.3 Đăng ký và sử dụng



❑ Ví dụ về gửi intent đến Broadcast Receiver

```
Intent intent = new Intent(NEW_EARTHQUAKE_FOUND);
intent.putExtra("date", quake.getDate().getTime());
intent.putExtra("details", quake.getDetails());
intent.putExtra("longitude", quake.getLocation().getLongitude());
intent.putExtra("latitude", quake.getLocation().getLatitude());
intent.putExtra("magnitude", quake.getMagnitude());
sendBroadcast(intent);
```



Nội dung



1. Broadcast Receiver

2. Service

- Khái niệm
- Vòng đời Service
- Khởi tạo Service



2.1. Khái niệm



- **Service:** là một trong những thành phần của Android, được chạy trên background thực hiện những công việc tính toán không đòi hỏi giao diện.
- **Service giống như Activity** thực hiện theo chu kỳ thời gian để quản lý sự thay đổi của trạng thái.



2.1. Khái niệm

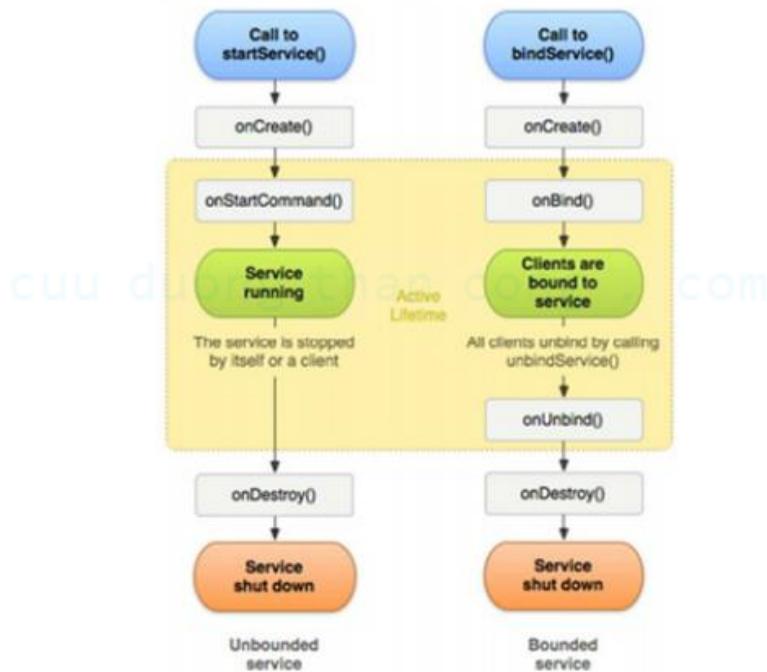


❑ Có 2 loại Service:

- “Started” Service: Service được gọi khi thành phần của ứng dụng gọi nó bằng phương thức startService(), và service sẽ chạy vô thời hạn. Loại service này thực hiện công việc đơn lẻ như download, tính toán và service sẽ dừng một khi công việc hoàn thành.
- “bound” Service: Service được gọi khi thành phần của ứng dụng gọi nó bằng phương thức bindService(). Loại service đề nghị giao diện Client – Server cho phép thành phần ứng dụng tương tác với nó, gửi yêu cầu, lấy kết quả. Một khi tất cả thành phần ứng dụng ngừng liên kết, Service kết thúc .



2.2 Vòng đời Service



2.3 Khởi tạo Service



❑ Có thể khởi tạo Service ở hai dạng:

- Service: Lớp cơ sở của tất cả Service
- IntentService: Lớp con của Service, là cách tốt nhất nếu không đòi hỏi Service handle nhiều yêu cầu đồng thời.



2.3 Khởi tạo Service



❑ Ví dụ khởi tạo IntentService:

```
public class HelloIntentService extends IntentService{  
    public HelloIntentService() {  
        super("HelloIntentService");  
    }  
    @Override  
    protected void onHandleIntent(Intent intent) {  
    }  
}
```



Thảo luận

